### *University of Pittsburgh*

### *School of Computing and Information*

**INFSCI 2711: Advanced Database Management System**

***Spring 2020***

**Final Project Report**

**A close up of a sign

Description automatically generated**

**Submitted By:**

|  |  |  |
| --- | --- | --- |
| **Name** | **Email Address** | **Student ID** |
| Aishwarya Jakka |  |  |
| Shruti Gupta | [shg104@pitt.edu](mailto:shg104@pitt.edu) |  |
| Piu Mallick | [pim16@pitt.edu](mailto:pim16@pitt.edu) | 4374215 |
| Reshma Sara Pothen | [rep83@pitt.edu](mailto:rep83@pitt.edu) |  |
| Soham Bhatnagar | [sob38@pitt.edu](mailto:sob38@pitt.edu) |  |
| Kenny Wu |  |  |
| Kwesi Randolph Aguillera | [kra40@pitt.edu](mailto:kra40@pitt.edu) |  |

**Online Retail Datawarehouse**

1. **Introduction**
2. **Sub-Teams**

Teams responsible for exploring various areas in the project:

* **Coordination**
  + Aishwarya Jakka
* **Initial Data Cleaning**
  + Kwesi Randolph Aguillera
* **Further Data Cleaning**
  + Respective Teams
* **SQL-DB**
  + Shruti Gupta
  + Aishwarya Jakka
* **MongoDB**
  + Piu Mallick
  + Reshma Sara Pothen
* **Neo4j**
  + Soham Bhatnagar
  + Kenny Wu
  + Kwesi Randolph Aguillera
  + Aishwarya Jakka
* **Front-end & Back-end integration**
  + Kenny Wu
  + Kwesi Randolph Aguillera
* **Documentation**
  + Piu Mallick
  + Reshma Sara Pothen
  + Shruti Gupta

1. **Dataset Description**

This [**Online Retail II**](https://archive.ics.uci.edu/ml/datasets/Online+Retail+II) data set (the original dataset is a **excel file**) contains all the transactions occurring for a **UK-based** and registered, non-store online retail between **01/12/2009 (1st Dec 2009)** and **09/12/2011 (9th Dec 2011)**. The company mainly sells **unique all-occasion giftware**. Many customers of the company are wholesalers.

* 1. **Attribute Information**

The dataset has the following attributes:

* **InvoiceNo**: Invoice number. Nominal. A 6-digit integral number uniquely assigned to each transaction. If this code starts with the letter 'c', it indicates a cancellation.
* **StockCode**: Product (item) code. Nominal. A 5-digit integral number uniquely assigned to each distinct product.
* **Description**: Product (item) name. Nominal.
* **Quantity**: The quantities of each product (item) per transaction. Numeric.
* **InvoiceDate**: Invoice date and time. Numeric. The day and time when a transaction was generated.
* **UnitPrice**: Unit price. Numeric. Product price per unit in sterling (Â£).
* **CustomerID**: Customer number. Nominal. A 5-digit integral number uniquely assigned to each customer.
* **Country**: Country name. Nominal. The name of the country where a customer resides.
  1. **Github Link**

The Github link for the project repository: [Click here](https://github.com/Aishwaryajakka/infsci2711_finalproject)

1. **Initial Data-Cleaning**

Following are the initial set of steps to clean the data:

* Removed records with blank CustomerIDs.
* Removed records weird StockCodes.
* Corrected Description where possible.
* Removed records where Description cannot be corrected.
* Removed zero Price items.
* Removed “Unspecified” country records.
* Changed Negative quantity records to positive quantity values.
* Separate Datetime into Date and Time columns.
* Calculate SubTotal for each record.

Following are the column stats for the year **2009-2010**:

* No. of records before cleaning: 525461
* No. of records after cleaning: 413084
* InvoiceNo Range: 489437 – 528618 / C489449 – C538168
* StockCode Range: 10002 – 90208 / 10123C – 90214Z & SP1002
* Quantity Range: 1 – 19,152
* Date Range: Dec 2009 – Dec 2010
* Time Range: 7:01 AM – 9:52 PM
* UnitPrice Range: $0.08 - $295.00
* CustomerID Range: 12346 – 18287
* No. of Countries: 36
* SubTotal Range: $0.06 - $15,818.40

Following are the column stats for the year **2010-2011**:

* No. of records before cleaning: 541910
* No. of records after cleaning: 300766
* InvoiceNo Range: 536365 – 564272 / C536383 – C564276
* StockCode Range: 10002 – 90208 / 10123C – 90214Z
* Quantity Range: 1 – 80,995
* Date Range: Dec 2010 – Dec 2001
* Time Range: 7:35 AM – 8:38 PM
* UnitPrice Range: $0.08 - $649.50
* CustomerID Range: 12347 - 18287

The **initial data cleaning** was done manually in **Microsoft Excel**.

Post initial data cleaning, some extra cleaning was done in order to meet various database needs, which will be discussed later in this report.

1. **Possible Aggregation Queries**

Following are the probable statistical and aggregation queries that the owner/manager of the store may want to view in order to get the growth report of their store:

1. What time of the day (which hour of the day) is the sale maximum per country?
2. What is the annual TotalSales per product?
3. What is the top product per year?
4. What is the top product per country?
5. Which item is sold below a certain threshold value? Or, what are the under-performed products based on the average sales last year?
6. Which customer spends the most (per country/overall)?
7. What is the best-selling month per country? (Given the year range, 2009-2011)
8. What is the best-selling product per month? (Given the year range, 2009-2011)
9. What is the change in TotalSales per country per year (Trend of Sales)?
10. What is the average spending of a customer per country? (TotalSales/Number of customers)
11. What is the frequently purchased item per customer?
12. **STAR Schema: Design**

Based on the queries (which we want to ask the database) stated above, we have designed the **DIMENSIONS** and **MEASURES**, which we popularly call as a **STAR schema**. Hence, first comes the aggregate queries, and based on the queries, we would like to build our aggregated database.

**DIMENSIONS**: Customer, Stock, Time

**MEASURES**: Quantity, Sales

**STAR SCHEMA:**

Hence, the **tables** (with all the **attributes**) we would be considering are as follows:

* **FACT** (*CustomerID, Year, Month, Day, Hour, Minute, StockCode, Quantity, TotalSales*)
* **CUSTOMER\_DIM** (*CustomerID, Country*)
* **STOCK\_DIM** (*StockCode, Description, MaxUnitPrice*)
* **TIME\_DIM** (*Year, Month, Day, Hour, Minute*)

  However, the above table definitions may change depending on the databases we would be using.

1. **Databases**

We have chosen 3 databases, namely **SQL, MongoDB** and **Neo4j** for building our **Online Retail Datawarehouse**. We would be integrating each of the databases with front-end, which we will discuss later in the document.

* 1. **SQL DB**

**7.1.1 Information goes here**

* 1. **MongoDB**
     1. **Creating the ODB (Operational Database)**
* **Extra data-cleaning for MongoDB:**

Steps followed to clean the **Cleaned2009-2010.xlsx** and **Cleaned 2010-2011.xlsx.** (actual data files) from the **data** folder:

* Changed the **file type** to **csv**.
* Loaded the **csv files** to **Jupyter Notebook** (**Python**) and converted them to dataframes.
* Concatenated two dataframes using the ‘**append’** function.
* The attribute ‘**Customer ID**’ is renames to ‘**CustomerID**’.
* The attributes ‘**InvoiveDate**’ and ‘**InvoiceTime**’ are then concatenated and renamed to ‘**InvoiceDateTime**’. Later, they are split to **Year** (**YYYY**), **Month** (**MM**), **Day** (**DD**), **Hour** (**HH**) and **Minute** (**MM**).
* The dataframe is then exported to the data folder as ‘**Online\_Retail\_DB.csv**’.
* **Steps followed to import the csv file to MongoDB:**
* Go to the **terminal** and login to **MongoDB** (by typing ‘**mongo**’ to stay in the same terminal or ‘**mongod**’ to switch to a new terminal).

**Pre-requisite**: Prior installation of MongoDB in the system and the mongodb service to be running.

* Switch to or create a new database – ‘**odb**’ in MongoDB with the help of the following command in the terminal:

**use odb**

* In a separate terminal, navigate to the path where the cleaned data file is present. The **MongoDB import command** is executed outside the mongo shell - in a normal terminal, with the help of the following command:

**mongoimport --db odb --collection retail --type csv --headerline --file Online\_Retail\_Data.csv**

* **Challenges Faced:**

The main challenge in ODB database was to split the InvoiceDate and InvoiceTime into individual Year, Month, Day, Hour, Minute (as the Date and Time were already split in the initial part of the data cleaning). Hence, we imported the csv file to Python to do the date and time splitting. Also, we appended two csv files to a single file. This could have also been achieved in the MongoDB as well. However, each import command in MongoDB (one for each csv file) was taking around 40 seconds. So, importing 2 csv files would have taken around 80 seconds. After combining the two files into one file, the import command took around 45 seconds, which means we saved around 35 seconds. Though, it is a matter of 35 seconds only. However, we are not sure if this strategy would work if any dataset is significantly larger than our dataset.

* + 1. **Creating the ADB (Analytical Database)**
    2. **Aggregation Queries**
  1. **Neo4j**
     1. **ODB Creation**
        1. **Creating CSVs**
        2. **Importing the Data**
        3. **Cleaning the ODB**
     2. **ADB Creation**
        1. **Creating Tables**
        2. **Creating the Relationships between the tables**
     3. **Aggregation Queries**
        1. **What time of the day (which hour of the day) is the sale maximum per country?**
        2. **What is the annual TotalSales per product?**
        3. **What is the top product per year?**
        4. **What is the top product per country?**
        5. **Which item is sold below a certain threshold value? Or, what are the under-performed products based on the average sales last year?**
        6. **Which customer spends the most (per country/overall)?**
        7. **What is the best-selling month per country? (Given the year range, 2009-2011)**
        8. **What is the best-selling product per month? (Given the year range, 2009-2011)**
        9. **What is the change in TotalSales per country per year (Trend of Sales)?**
        10. **What is the average spending of a customer per country? (TotalSales/Number of customers)**
        11. **What is the frequently purchased item per customer?**
     4. **Justifications**
     5. **Advantages of Neo4j**
     6. **Disadvantages of Neo4j**
  2. **Front-End**

1. **Comparison**
2. **Conclusion**